AMYGDALA $z \mapsto z^2+c$

A Newsletter of fractals & $\mathcal{M}$ -- the Mandelbrot Set
AMYGDALA, Box 219, San Cristobal, NM 87564

$15.00 for ten issues ($25 overseas)
$15.00 for slide supplement ($25 overseas)
$30.00 for ten issues plus slides ($40 overseas)

Issue #4
May 1987

CONTENTS

THE LOGO

The masthead is changed to include a little picture of $\mathcal{M}$ with the interior detailed. We also incorporated this logo into the design for the envelopes this issue was mailed in.

Perhaps you thought that the interior of $\mathcal{M}$ had to be all black — so where does the detail come from? For an explanation, see "Exploring the Interior of $\mathcal{M}$", below.

EXPLORING THE INTERIOR OF $\mathcal{M}$

The idea for this detailing of the interior of $\mathcal{M}$ comes from Figure 33 in "The Beauty of Fractals", page 60.

For each point z we perform the usual iteration: $z_0 = 0$, $z_{n+1} = z_n^2 + z$, but we also compute the minimum value of $|z_n|$ as we go — call it **inf(z)**. Figure 33 of TBF shows the levels of inf(z) (for z in $\mathcal{M}$) in alternating black and white stripes.

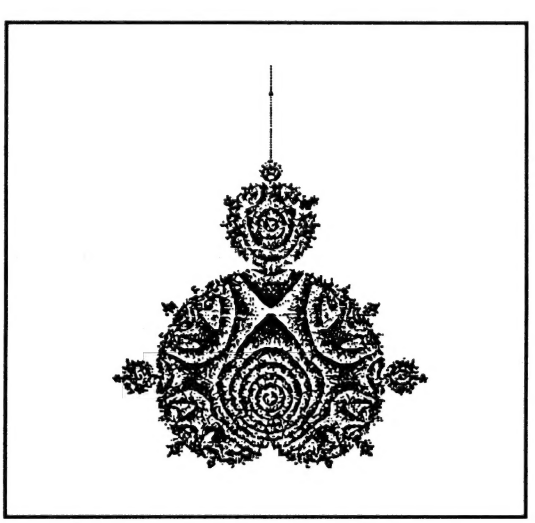
For our logo we use the same function inf(z), but do something slightly different with it, to get the effect of a gray-scale shading with 65 different gray levels, running from all white through 63 gray levels to all black. First we arbitrarily choose a black level B — the smallest value of the modulus which will be colored black. For each z in $\mathcal{M}$ we convert inf(z) to a gray level between 0 and 64 as follows: $k = \lfloor 64 \cdot \text{inf}(z)/B \rfloor \bmod 65$ is an integer between 0 and 64, which achieves 64 (black) for the first time when inf(z) = B, which is why B is called the black level. We then shade the pixel associated with z with gray shade k.

For the logo we chose B = 0.041, and turned the resulting figure through 90°. The picture to the right uses the same algorithm, but with B = 0.18. In both cases, the iteration limit was taken to be 511.

Details of how the gray-scale shading is done will be discussed in a future issue. We use **pattern shading**.

NOTATION

There are many ways to describe the frame (rectangular window) in which a detail of $\mathcal{M}$ is displayed: coordinates of lower left / upper right corners, lower left and dimensions, etc. I like to use a notation based on three numbers: (1) the complex number representing the point at the center of the rectangle, (2) the "magnification" 1/R, where R is the radius of the largest circle that will fit inside the rectangle, and (3) the pixelation XxY.



For example, the colored image shown in John Dewey Jones's slide which was included with Amy #2 has -1.99641-0.000024i as its lower left-hand corner and -1.99635+0.000024i as its upper right-hand corner. Therefore the center is -1.99638 and the magnification is 1/0.000024 = 41,667.

The center should be given to a precision sufficient to specify it to within 1%(say) of the radius of the picture.

As to pixelation, John wrote: "The image was photographed (1 sec. at f8) from the screen of an IBM AT/G. The AT/G screen is 720 pixels wide by 500 pixels high; however, I've used 2 horizontally adjacent pixels for each point in the complex plane in order to extend my palette from the 8 colors available per pixel to a possible 32 per pixel pair." I'm not sure how to boil this down to a simple XxY specification of pixelation. Is it 720x500? 360x500? The "aspect ratio" of the picture is .000060/.000048 = 1.25, while that of the screen is 720/500 = 1.44. Are the pixels square? Does the picture not fill the whole screen? Was it stretched horizontally?

TERMINOLOGY

The iterates of a complex number c are the numbers $z_0 = 0$, $z_1 = c$, $z_2 = z_1^2+c$, $z_3 = z_2^2+c$, ... There's a choice here: Should we start with $z_0 = 0$ or c? Is the choice arbitrary? I call the largest n for which $|z_n| \leq 2$ the **dwell** of c. If all $|z_n| \leq 2$, i.e. $c \in \mathcal{M}$, then the dwell is infinite. If $|c| > 2$, the dwell is 0.[1]

TUTORIAL: FUNCTIONS; POLYNOMIALS

I'm going to address those of you that have some background in functions of a real variable, and are curious as to how that knowledge extends to the case of the complex numbers.

The concept of a complex-valued function of a complex variable parallels that of a real-valued function of a real variable. Such a function f associates with each complex number z in the domain D of the function another complex number f(z) in its range.

The definition of continuity for complex functions parallels that of real functions, as does that of differentiability. A single-valued function f is called *analytic* in a domain D if it has a derivative at every point in D. Unlike their real counterparts, analytic functions have a remarkable property: merely having a first derivative insures the existence of all higher derivatives.

Complex polynomials are functions defined analogously to their real counterparts. Associated with n+1 complex numbers $a_0,...,a_n$, with $a_n \neq 0$, is a polynomial P: a function for which $P(z) = a_n z^n +...+ a_0$. Any polynomial is analytic over the whole of $\mathbb{C}$.

[1] I have dithered back and forth between $z_0 = 0$ and $z_0 = c$, but now propose to settle on the former.

DR. CARL SPRING REPLIES

27 April 2087

From: Carl Spring
Executive Director
Campaign for a Real World

Dear Mr. Silver,

I hope you will grant me the right of reply in your columns to the attacks on me, and the half-truths about the Object, perpetrated by my former colleague Dr. Charles Festus in Amygdala #3.

Much as Dr Festus's defamation of me and deliberate misrepresentation of my motives calls out for rebuttal, the Object comes first. Let me say, then, that Dr. Festus's article completely misrepresents the nature of our discovery — and he misrepresents it, moreover, from the basest of motives: from the desire to add yet further to the profits of his company, Fractal Growth, no matter what the risk to the public.

Your readers may have noticed that, despite the plethora of detail Festus provides on obsolete accelerators — detail that could only be of interest to an industrial archæologist — he never really explains the *physics* of what happened that July day in 2052. Yet he knows what happened well enough; he was there when Paul and I completed our analysis of Susan's data. But by that time he'd already begun thinking of the money to be made. And within a few years he was making it — enough money to buy Paul and Susan's silence, to keep my warnings out of all but the most obscure journals, and to cover up the disappearances of the unfortunate few who never returned from his 'Fractal Safaris'.

I know what the media said of me — that it was just sour grapes, or, as the Post put it, 'bitter almonds'. Well, yes, I am bitter — not so much that I never received proper credit for my discovery, but that that discovery's true nature is being dangerously misrepresented for one man's profit.

From Festus's account, you'd think the Object really existed. I've seen it, and I know it doesn't. Those who first described the Object last century know what it was, and where it was to be found: in the complex plane, where real and imaginary axes meet. Well, the world of our experience is real, wholly real, only a solipsist or a mystic would deny that; there's no room in it for imaginary or complex entities.

But in trying to relate our experimental results to established physics, Paul and I were led to posit that adjacent to the real world there are many imaginary worlds, 'shadow worlds', we called them, whose physical properties are representable by imaginary magnitudes. These 'shadow worlds' are analogs of the real world, just as antimatter is an analog of normal matter. What we'd done with our 2 EeV collision was to open a tiny aperture into an imaginary world; the Object appeared in the plane where real and imaginary worlds

met. And it was very fortunate for us that it did; without the Object standing there as a gatekeeper, imaginary entities could have leaked through into the real world. Or, worse still, one of us might have passed through the gateway into the imaginary world.

Now let me stress this: any talk of travelling *within* the Object is plain nonsense. It's a two-dimensional phenomenon, and one of those two dimensions is imaginary. What, then, happened to Sind Balland? And where do the Fractal Growth safaris actually take their clients — their victims, I should say? Even after we'd analyzed the physics of the Object's sudden appearance, it took me several years to understand this — years during which we, like the discoverers of radium, played innocently with our new toy, ignorant of the damage we were doing to ourselves. Yes, I went along on Festus's joyrides, thinking that we were witnessing the wonders of fractal space. Well, we weren't. And to continue selling this myth, when we know what's really going on, is criminally irresponsible. You hear me, Festus: *criminal!*

Either because the Object does not perfectly seal the gateway with the imaginary world, or owing to some form of quantum tunneling, there is a flux of imaginary particles across the gateway, and a corresponding leakage of real matter into the imaginary world. Exposure to this particle flux naturally acts as a powerful stimulant to the imagination, while depressing the rational and logical functions of the brain. Under these conditions the most vivid and convincing hallucinations occur, their content suggested by the numinous apparition of the Object — helped along, these days, by Dr. Festus's publicity handouts and suggestive props. But we paid a price for these hallucinations. Each of our encounters with the Object left us progressively drained of reality; imaginary particles increasingly displaced the physical constituents of our bodies. Sind Balland, whose first accidental trip took him closer to the gateway than anyone has since ventured, was reduced to a grey and flickering shadow. He now lives in a rest home, with a generous pension from Fractal Growth to keep him quiet. The damage we must have done to ourselves was more insidious. Though I have not approached the Object for twenty years now, I still have flashbacks of unreality — the last of which must have occasioned Dr. Festus's wishfully premature announcement of my death. I like to believe that it's the continuing exposure to unreality that has allowed Paul and Susan to go along with Festus's schemes, rather than the profitability of their shares in Fractal Growth. In Festus's own case, I'm afraid that the first of his faculties to lose its reality was his conscience.

There's Only One Reality — Let's Keep it Safe! Close the Gateway to the Imaginary World!

Yours faithfully,

Carl Spring

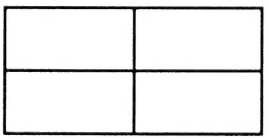
—John Dewey Jones

MARIANI'S ALGORITHM

Rico Mariani has discovered a powerful algorithm for speeding up the calculation of views of the Mandelbrot set. He describes it as follows:

About my algorithm: it really isn't very amazing, and it has a few drawbacks (such as requiring that a copy of the entire picture be kept in memory during the computations, and a large queue for rectangular regions... but I'm getting ahead of myself).

Here's the idea: Whenever you calculate M or a subset of M, the computer tends to spend most of its time computing these large chunks of solid black area, i.e. points that are in M and hence required the full number of iterations to compute. My idea is to try to compute as little as possible of these areas.



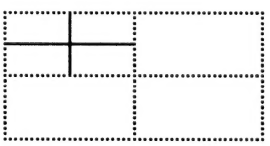
Here's the algorithm:

Take the area of interest and compute the pixels as shown in the diagram (i.e. all around the perimeter, plus a vertical and horizontal line). You must ensure that the M-set is intersected by at least one of these lines. This divides your region into four rectangles.

Now check the perimeter of each rectangle:

If every pixel on the perimeter required the same number of iterations before it failed the test (e.g. every pixel is a member of M) then don't compute any pixels inside the rectangle. Just consider them to all have the same iteration count as the ones on the perimeter (e.g. if you find any rectangle which has every point on its perimeter in M then the whole rectangle is in M).

If all the pixels do not have the same number of iterations, then divide the rectangle into four sub-rectangles by drawing horizontal and vertical lines though it... giving a pattern of computed pixels something like this:



Repeat this procedure on the largest remaining rectangle (this happens automatically if you put your rectangles into a queue as you create them).

You bottom out the recursion that I just described when the rectangles get suitably small (how small is up to you). When they're "small enough" (say smaller than 5 by 5) you compute all of the pixels in the rectangle (avoid recomputing the perimeter if you can, this will save much time).

Why is this faster than the usual way? Well... whenever there are large areas which are the same colour (i.e. have the same number of iterations), you'll tend to get rectangles fitting completely inside this region. When this happens you save time because the interior of the rectangle need not be computed, only the perimeter. If the rectangle is completely in M this can save MANY computations!

Caveats: the algorithm isn't perfect... if the algorithm could guarantee that the whole perimeter of a rectangle was the "same colour" then it would always work, however since

it only makes finitely many samples on the perimeter we can't really be sure. That means that the inside might not really be the same color as the perimeter. In practice the algorithm gives good results and is definitely suitable for use as a "fast preview" type thing. Also, as I've noted, it needs to keep this rectangle queue around, and it has to remember how many iterations were required for each pixel... These can be expensive in terms of storage.

It gives the best speed improvements when the display has a lot of points that are in M. Try it out on regions with a side of length .01 or .001... it works pretty good at that magnification.

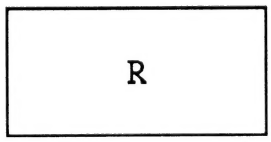
MARIANI'S ALGORITHM MODIFIED

The essential idea of Mariani's algorithm is to partition the field of view into rectangles, subrectangles, sub-subrectangles, etc., noting that if the perimeter of any rectangle is composed of points all having the same dwell, then all points within that rectangle have that same dwell, and no iterations need to be performed on them. Since most images of interest have broad areas of constant dwell, this method can be applied to decrease the amount of computation involved.

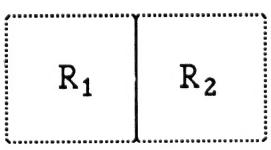
The algorithm as presented above is unnecessarily restrictive in some of its details and assumptions. Rather than noting these piecemeal I have decided to paraphrase the algorithm, incorporating a few changes which will, I hope, improve it.

For example, by employing a depth-first recursion, it is not necessary to maintain a queue of rectangles. For another, the initial configuration can be a simple unquartered rectangle, and the requirement that M intersect at least one of the initial lines can be dropped in favor of a simpler one: that not all of $\mathcal{M}$ be included in the rectangle.

Take a rectangular area R of interest, which does not include all of $\mathcal{M}$, and compute the dwell of all pixels around the perimeter of R (which, for brevity, we call ∂R). Either all of these dwells are the same, or not. If they are all the same, we have a rather uninteresting picture: all the pixels have that same dwell, within R or on its perimeter. Just color the whole of R mauve, or whatever.



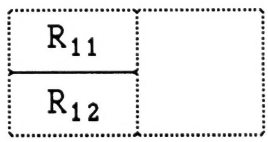
For an interesting picture, the dwells of pixels on ∂R are not all the same. Proceed by dividing the interior of R into two approximately equal sub-rectangles with a line parallel to the shorter side. In the following figure, ∂R is shown dotted, indicating that the dwells of its pixels have already been computed.



Now compute the dwell of those pixels of ∂R1 and ∂R2 which have not yet been computed: those lying on the solid line in the previous figure. Then process each of the subrectangles R1 and R2 in turn.

Just as with R itself, the dwells of all pixels on ∂R1 are either the same, or they are not. If they are all the same, all the pixels of R1 have that same dwell, within R1 or on its perimeter, and those in the interior should be assigned that dwell, without any further iterative calculations being performed.

If not all pixels of ∂R1 have the same dwell, repeat the procedure on the two sub-subrectangles of R1: →



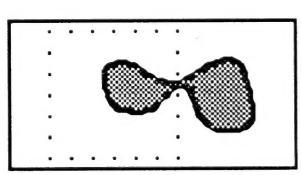
Compute the dwell of those pixels of ∂R11 and ∂R12 which have not yet been computed: those lying on the solid line in the figure. Then process each of the subrectangles R11 and R12 in turn.

Stop the recursion (partitioning into two subrectangles) when a rectangle is encountered having either (1) constant dwell on its boundary, or (2) empty interior, i.e. it is of dimensions 1xn, nx1, 2xn, or nx2.

In all cases, avoid recomputing any pixels already computed.

Whenever there are large areas in which all pixels have the same dwell, you'll tend to get rectangles fitting completely inside this region. When this happens you save time, because the interior of the rectangle need not be computed, only the perimeter.

The algorithm isn't guaranteed to work. It is true that if the dwell of all points on ∂R are the same (and R doesn't contain all of $\mathcal{M}$) then all points within R have that same dwell. But we are only approximating this with a finite number of points on ∂R. It is possible for a filament of another color to sneak through between two points on ∂R, and expand within R to a significant region, e.g., as in the figure to the right.



In practice, this is not likely to occur.

"EAT MY DUST!"

Mariani's algorithm is essentially based on a "monochromatic"[2] theorem about the iteration $z \mapsto z^2+c$: Let R be a region in the complex plane not containing all of $\mathcal{M}$, for which all points z on the boundary ∂R have constant dwell d $\leq \infty$. Then all points in R have dwell d. A proof of this theorem will be given in the next issue.

[2] The term "monochromatic" was suggested by John Dewey Jones.

My modification of Mariani's algorithm, described above, improves it, I think. To avoid confusion I have named the modified algorithm "DivCon", for "Divide & Conquer".

Given a discrete set R of points in the complex plane $\mathbb{C}$ arranged in a square array, and not containing all of $\mathfrak{M}$, DivCon computes the dwells of all points in R as follows:

(1)If R consists of a single point, or a single row of points, compute all their dwells; otherwise:

(2)Compute the dwells of all points on the boundary ∂R;

(3)If all points in ∂R have the same dwell d, then all points in R have dwell d; otherwise:

(4)All points in ∂R do not have the same dwell. Subdivide R into two approximately equal parts, R1 and R2, using a straight line L parallel to its shorter sides. Compute the dwell of all points on L whose dwell has not been already computed.

(5)Apply steps 3-5 recursively to each of R1 and R2.

Note that when (3) is applied recursively to a subrectangle S, the dwells of all $z \in \partial S$ have already been computed.

I have used DivCon to create a piece of Mac shareware which I call "Eat My Dust!" (EMD) because of its speed.

The current version of EMD (EMD2) uses the Mac interface, but has only black and white shading of dwell zones. It will be upgraded to accomodate gray-level shading, full user control over dwell zone shading, large, scrollable pictures, arbitrarily large magnification...

Readers can obtain EMD2 on a single-sided diskette by sending $15.00 to Amygdala. EMD2 is **shareware**, which means that you can copy it and give it away. Anyone who receives a copy and wants to keep it should likewise send $15.00 to Amygdala, and I will register them as users. Users will get one free update, and notices of further updates.

EMD2 acquires its speed from two sources.

First, its inner loop is written in Assembler, using a 32 bit fixed-point format and exploiting the 68000's native 16x16 bit multiply. This format was discussed in Amy #1, page 5: it has 29 fraction bits, with negative numbers represented as the 2-s complement of their positive counterparts, and is sufficient to represent numbers in the range: $-(4-2^{29})...4-2^{29}$, which is adequate for our purposes so far. For comparison, programs exploiting the extended-precision format available in such chips as the Intel 8087 and the Motorola 68881 have 64 bits of precision.

Second, EMD2 gains its speed by using DivCon to evade computing the dwells of many points.

An idea of the speed of EMD2 can be gained from considering its performance on Mark Bolme's benchmark (Amy #1, page 4): calculate points in the rectangle with center -0.744353+0.113454i and magnification 200,000. All these points have dwell 50. Mark gives the following performance figures for an 8 MHz IBM PC clone:

with 8087 chip: 94.0 points per second;

without 8087: 6.24 points per second.

To provide a direct comparison with these results, I constructed a version of EMD which uses the fast dwell routine, but not the DivCon algorithm. This "TV scan" EMD achieved a speed of 129 points per second: 37% faster than the PC clone **with** the math chip. Another figure that can be derived from this test is that the fast dwell routine computes about 6450 iterations per second.

When DivCon was unleashed, the full EMD2 program computed the dwells of 251x251 points in 9 seconds, for an effective speed of 7000 points/second!

I admit that this doesn't have much practical significance, but a benchmark is a benchmark! I'll publish performance figures for EMD2 on a number of Mandelbrot views in the next issue, to serve as a basis for a realistic appraisal of its performance.

LETTERS

From Mark Bolme:

Dear Rollo:

Thanks for the look at your program! The speed of the algorithm was impressive and worthwhile even though it is not air-tight.. I have assumed that your theorem is correct without checking, **but** the program doesn't calculate **every z on the boundary of R**, merely the z's that lie on the grid of pixels. Z's that fall "in between the pixels" are not calculated.

I have visually inspected some of my pictures calculated the "old fashioned way", and found several places where points with dwell < limit were completely surrounded by points with dwell = limit (this was the quickest test I could think of). I have no doubt that there was a connection between these points that snaked out between the pixels, but the points themselves would be missed by your algorithm, unless they happened to fall on the edge of the rectangle.[3]

Nevertheless, your algorithm certainly seems useful, at least as a "quick look" option (that might be used 99% of the time).

Unfortunately, I see that you are marketing your program. This makes us competitive, a situation that I had hoped to avoid. You might want to reconsider publishing a newsletter **and** a program (MacUser magazine publishes no programs for instance). If you decide to continue, it would be difficult for us to proceed with our plans to let our users know of your newsletter. Let me know what your feelings are.

Good work on the program!

...

I have done a little benchmarking that you might be interested in... [Mark then reports that for the default included with the program (the whole Mandelbrot set) DivCon is 55% faster than linear scan, but when the program is applied to the picture shown in Scientific American, August 1985, page 18, figure d, the speedup is only 8.3%.]

---

[3] Comment by RS: The critical question is whether there are **rectangles** for which the algorithm fails, rather than just "places".

From Dennis Cutburth:
Great slide. I had it enlarged to an 8x12 color print. Very nice. Sign me up for the 25 slides for $15.
The math tutorial is excellent! Mandelbrot's book is over my head (but beautiful). Your tutorial makes this topic much more accessible. I think it probably helps extend the interest in the field by helping people understand what these pretty pictures are that they are looking at. It you can continue to explain these topics at a level a first or second year student can understand I think you will be able to reach a fairly large audience.

From Tom Granvold:
A couple of nights ago I went to a talk given by Heinz-Otto Peitgen, one of the authors of The Beauty of Fractals. Most of what he spoke of is in that book, as were most of the slides that he showed. Of particular interest were two short movies that he showed.
The first was a "flight" through an area around the area of the M Set he calls the Seahorse Valley. Only forty images of the set were computed for this film. The remaining frames were created by interpolating between the forty images.
The second film was the more interesting of the two. In one corner of the film was an outline of half of the M Set where a point would move along it, changing color as it went. In another corner was a blowup of the area where the moving point was. The rest of the screen showed the corresponding Julia Set for the position of the moving point. It was amazing to watch how the Julia Sets would flow from one form to another and how this form would be similar to the corresponding area of the M set. If you get a chance to attend a talk by Peitgen, this film alone makes it worthwhile to go.
He also mentioned a new algorithm that can be used to find the boundary of the M Set. He will present this algorithm this summer at the SIGGRAPH convention in Anaheim (I think that this is where it will be held) during a tutorial on fractals. Currently he is teaching at the University of California at Santa Cruz. He is an interesting speaker and well worth going to hear.

BIBLIOGRAPHY

44. P Sorenson, "Fractals". *Byte* 9 (1984) 157-172.

45. C Pickover & A Khorasani, "Fractal Characterization of Speech Waveform Graphs", *Computers and Graphics* 10 (1986) 51-61.

46. R Rivlin, "Computer Graphics: the Arts", *Omni Magazine* 8 (1986) 30.

47. C Pickover, "Blooming Integers", *Computer Graphics World* 10,3 (March 1987) 54-57. ["An elegantly simple algorithm generates complex patterns."]

48. C Pickover, "Mathematics and Beauty: Time-Discrete Phase Planes Associated With the Cyclic System {dx/dt = -f(y(t)), dy/dt = f(x(t))}", *IBM Research Report RC 11971* (1986). [Available as RC 11971 from IBMDS (IBM Distribution Services.]

49.MF Barnsley & SG Demko (eds), "Chaotic Dynamics and Fractals". Academic Press (1986)

50.BB Mandelbrot, "Fractals: Form, Chance and Dimension". WH Freeman (1977). [This essay is followed by, and largely replaced by, *The Fractal Geometry of Nature*].

51.I Peterson, "Zeroing in on Chaos". *Science News* (February 28, 1987) Front cover, 28-30 [Discusses the work of SA Burns, HE Benzinger & JI Palmore at the University of Illinois, studying the application of Newton's method to rational functions. Several interesting pictures, including a four-color (plus black) cover picture of the domains of attraction of the four roots of z^4-1].

AUTHORS

Dr. John Dewey Jones is a senior research engineer with the Engine Research Department of General Motors Research Laboratories, Warren, MI 48090-9055. His research interests include adiabatic diesel engines, thermodynamics, and the use of mathematical and computational methods in engineering.

CIRCULATION

As of today (May 31, 1987) Amygdala has 145 paid-up subscribers, of which 29 have the supplemental color picture subscription.